**Implementing jQuery AJAX with Entity Framework in ASP.NET MVC**

This tutorial helps you to implement **jQuery AJAX** in your **ASP.NET MVC**project. You will also learn how to use**Entity Framework** in doing database operations.

This tutorial is for Beginners and Intermediate coders who want to learn jQuery AJAX and Entity frameworks.

\*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\*

***To understand the jQuery AJAX function, please have a look to the “***[***jQuery AJAX***](http://www.yogihosting.com/jquery-ajax/)***” article for more details.***

\*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\*

This tutorial will explain them the following things –

1. How to use jQuery AJAX in ASP.NET MVC.

2. How to use Entity Framework for doing DB operations.

3. How to apply jQuery Validations in the View.

4. How to apply CSS in the view.

**Requirements**

I have used **ASP.NET MVC 5.0 and Entity Framework 6.0** in this tutorial. However the codes will run on all MVC and EF versions.

The Database is SQL Server 2012.

Visual Studio 2015 is used.

**Project Briefings**

In this project I will make a Database table in SQL Server. Then I will create a MVC form, on submitting this form, the form’s data will be sent to a function in the Controller using jQuery AJAX. This function will do the insert in the DB table using Entity Framework and return the outcome of the operation.

**STEP 1 — Creating the Database Table.**

Let us first create a database table containing the student’s information. It will have the following fields.

*a.****Id****— INT — Identity — Primary Key*

*b.****Name****— Varchar(50) — Not Null*

*c.****Address****— Varchar(50) — Allow Null*

*d.****Age****— Int — Not Null*

*e.****Standard****— Varchar(10) — Not Null*

*f.****Percent****— Decimal(5,2) — Not Null*

*g.****AddedOn****— Datetime — Not Null — default binding getdate()*

*h.****Status****— Bit — Not Null*

A screenshot of a data

Description automatically generated

**Database Table**

You can run the below **script**in your database to generate the Student’s table:

SET ANSI\_NULLS ON  
GO  
  
SET QUOTED\_IDENTIFIER ON  
GO  
  
SET ANSI\_PADDING ON  
GO  
  
CREATE TABLE [dbo].[Student](  
 [Id] [int] IDENTITY(1,1) NOT NULL,  
 [Name] [varchar](50) NOT NULL,  
 [Address] [varchar](100) NULL,  
 [Age] [int] NOT NULL,  
 [Standard] [varchar](10) NOT NULL,  
 [Percent] [decimal](5, 2) NOT NULL,  
 [AddedOn] [datetime] NOT NULL,  
 [Status] [bit] NOT NULL,  
 CONSTRAINT [PK\_Student] PRIMARY KEY CLUSTERED   
(  
 [Id] ASC  
)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]  
) ON [PRIMARY]  
  
GO  
  
SET ANSI\_PADDING OFF  
GO  
  
ALTER TABLE [dbo].[Student] ADD CONSTRAINT [DF\_Student\_AddedOn] DEFAULT (getdate()) FOR [AddedOn]  
GO

There will also be an Insert Stored Procedure ***“sp\_InsertStudent”*** which will insert entries in the student’s table. Note that this SP will be executed with the Entity Framework.

The SP code is given below:

CREATE PROCEDURE [dbo].[sp\_InsertStudent]   
 @Name VARCHAR(50),   
 @Address VARCHAR(100),   
 @Age INT,  
 @Standard VARCHAR(10),  
 @Percent DECIMAL(5,2),  
 @Status bit,   
 @Result VARCHAR(50) OUTPUT,  
 @CreatedId INT OUTPUT   
AS   
BEGIN  
 INSERT INTO Student(Name,Address,Age,Standard,[Percent],Status) VALUES (@Name,@Address,@Age,@Standard,@Percent,@Status)   
 SET @Result='Insert Successful'   
 SET @CreatedId=@@IDENTITY  
END

**STEP 2 — Creating the Entity Data Model (EDM)**

Now let’s move towards the Entity Framework. I will create **EDM**model of the Database, this EDM will create a **.edmx** file.

The **EDMX**file contains the knowledge of the database table, and the insert stored procedure.

Because of this EDMX file, we will be able to execute the ***“sp\_GetStudent”*** SP from the MVC code.

Follow the Below Steps to create EDM:

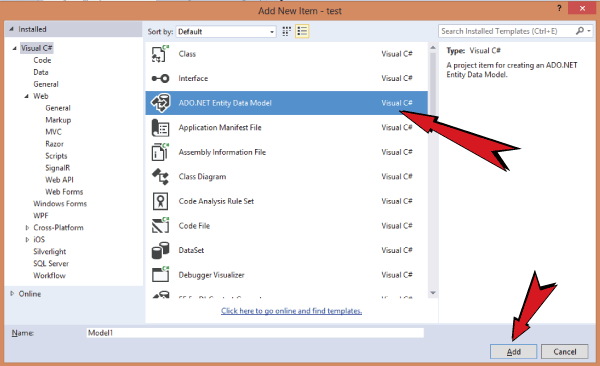
***a. Install “Entity Framework” from NuGet Package****— Ignore this step if EF is already installed in your Project.*

In the Visual Studio right click the project then select “Manage NuGet Packages” . Next, in the search box, enter “Entity FrameworK” and click enter.

Select the “Entity Framework” and click “Install” button.

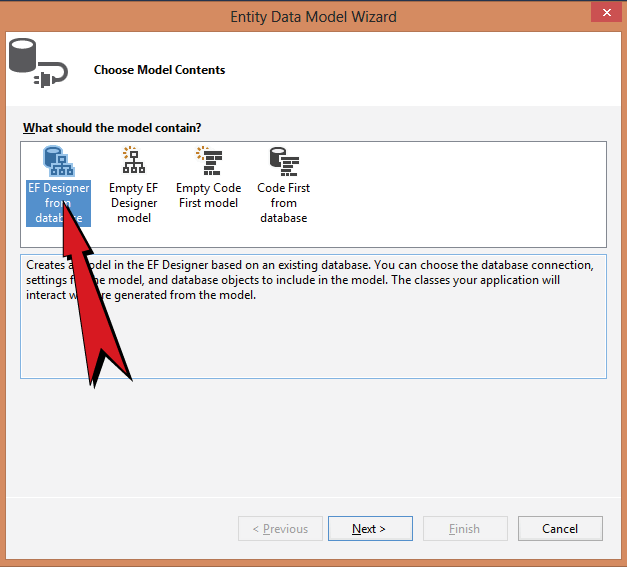
***b. Add “ADO.NET Entity Data Model”****— In the Visual Studio right click the project. Then select “Add > New Item” — this will open “Add New Item” Window.*

Next select “Visual C#” from the left side (under “Installed”), then select “**ADO.NET Entity Data Model**” and click the “Add” button.



**ADO.NET Entity Data Model**

***c. Choose Model Contents****— Select “EF Designer from database” and click next button.*



**Choose Model Contents**

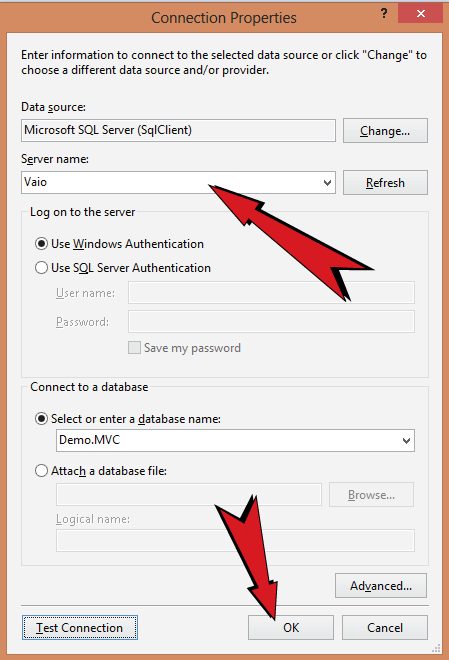
***d. Choose Your Data Connection****— Click the “New Connection” button to open the “Connection Properties” window.*

Enter the “**Server name**” and then select the database from “**Select or enter a database name**”. Next click the “**OK**” button.

The “**Connection Properties**” window will close.

Click the “**Next >**” button.

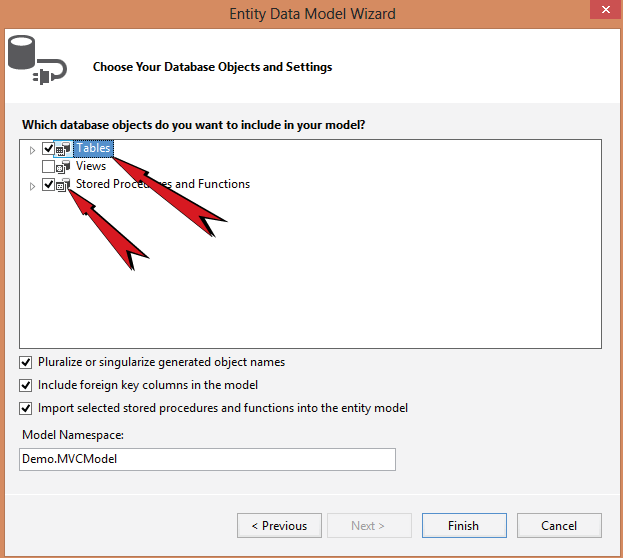
**Note** — select the database where you have created the “**Student**” table. Use the “**Test Connection**” button to check if the connection is alright or now.



**Choose Your Data Connection**

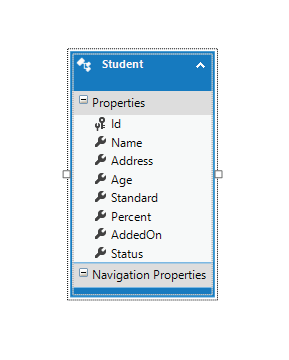
***e. Choose Your Database Objects and Settings****–Check the “Tables” and “Stored Procedures and Functions”. This is because I have one table and one stored procedure and I want the EDM to have the knowledge of both of them.*

Finally click the “Finish” button.



**Choose Your Database Objects and Settings**

Finally the **EDMX file** will be created and it will have the students table shown over it.

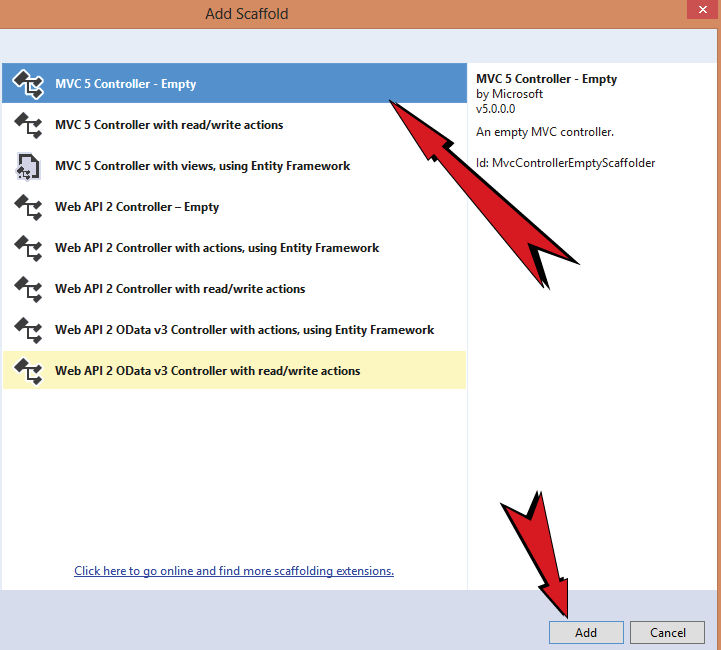


**EDMX file**

**STEP 3 — Creating the Controller**

Right click the “Controller” folder in your project and select “Add > Controller”. Next Click “**MVC 5 Controller — Empty**” and click “Add” button.

Give the Controller name as “**InsertController**” and click the “**Add**” button.



**Creating the Controller**

The controller will open in the Visual Studio. Now start entering the code.

First add the entity namespace to the controller-

using System.Data.Entity.Core.Objects;

Then create an insert function in the controller. This function will call the “**InsertStudent**” stored procedure. Notice how I have defined the **2 output parameter**of the SP as type “**ObjectParameter**”.

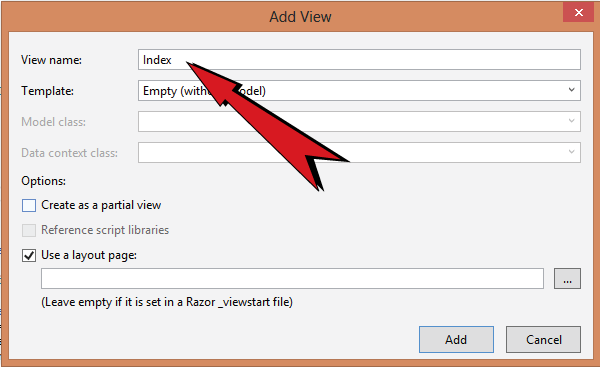
This function — I will call through the**jQuery AJAX** method from the view. I will explain how to do this in STEP 7.

The values of the “**result**” output parameter is returned at the last line.

[HttpPost]  
public string Insert(string name, string age, string standard, string percent, string address, string status)  
{  
 using (var context = new Entities())  
 {  
 ObjectParameter result = new ObjectParameter("result", typeof(String));  
 ObjectParameter createdId = new ObjectParameter("createdId", typeof(String));  
 context.sp\_InsertStudent(name, address, Convert.ToInt32(age), standard, Convert.ToDecimal(percent), Convert.ToBoolean(Convert.ToInt32(status)), result, createdId);  
 return result.Value.ToString();  
 }  
}

**STEP 4 — Creating the View**

In your project open the “Views” folder where you will see the “**Insert**” folder. Right click the “Insert” folder and select “**Add > View**”. Give the view name as “**Index**” and click the “**Add**” button. This will create the view.



**Creating the View**

**STEP 5 — Creating the Student Form in the View**

Now I will create the “**Student**” form with all the fields given in the “Student” **table**. I will not include the “**Id**” and “**AddedOn**” as they are auto-generated fields.

The below code will create the student form in the “**Index**” view:

<div id="messageDiv"></div>  
<div class="studentFormDiv">  
 @using (Html.BeginForm())  
 {  
 <table>  
 <tr>  
 <td>  
 <label>name</label>  
 <input id="name" type="text">  
 <span class="nameSpan"></span>  
 </td>  
 <td>  
 <label>age</label>  
 <input id="age" name="age" type="number">  
 <span class="ageSpan"></span>  
 </td>  
 </tr>  
 <tr>  
 <td>  
 <label>standard</label>  
 <input id="standard" type="text">  
 <span class="standardSpan"></span>  
 </td>  
 <td>  
 <label>percent</label>  
 <input id="percent" type="text">  
 <span class="percentSpan"></span>  
 </td>  
 </tr>  
 <tr>  
 <td>  
 <label>address</label>  
 <input id="address" type="text">  
 <span class="addressSpan"></span>  
 </td>  
 <td>  
 <label>status</label>  
 <select id="status">  
 <option value="Select">Select</option>  
 <option value="1">Active</option>  
 <option value="0">InActive</option>  
 </select>  
 <span class="statusSpan"></span>  
 </td>  
 </tr>  
 <tr><td><button id="submitButton" type="submit">Submit</button></td></tr>  
 </table>  
 }  
</div>

**Explanation**— The above form will be filled and submitted by clicking the submit button at the bottom. On clicking the button the **jQuery AJAX**will call the C# function “**Insert**”. The returned value from the C# function will be shown in the “**messageDiv**” div.

**STEP 6– Applying the Styles to the Form**

To give the little bit of styling to this form add the **CSS**to the view, on top of the form:

<style>  
 .studentFormDiv label {  
 display: block;  
 margin: 0;  
 text-transform: capitalize;  
 }  
  
 .studentFormDiv span {  
 display: block;  
 color: red;  
 }  
  
 .studentFormDiv > form > div > div {  
 padding-top: 25px;  
 }  
  
 table {  
 width: 100%;  
 }  
  
 h1 {  
 border-bottom: solid 3px #CCC;  
 }  
</style>

**STEP 7 — Using jQuery AJAX to Call the Insert Function**

On the “**submitButton**” button click event I will validate all the form’s fields with jQuery.

When all the field values are correct then I will use jQuery AJAX method to call the C# function “insert”.

The student’s field values will be sent to this function as parameters. Since this function will be called using jQuery AJAX so there will be no page postback.

The jQuery Code is given below:

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.0/jquery.min.js"></script><script>  
 $(document).ready(function () {  
 $("#submitButton").click(function (e) {  
 var returnValidate = Validate();  
 if (returnValidate) {  
 $.ajax({  
 type: "POST",  
 url: "/insert/insert",  
 contentType: "application/json; charset=utf-8",  
 data: '{"name":"' + $("#name").val() + '","age":"' + $("#age").val() + '","standard":"' + $("#standard").val() + '","percent":"' + $("#percent").val() + '","address":"' + $("#address").val() + '","status":"' + $("#status").val() + '"}',  
 dataType: "html",  
 success: function (result, status, xhr) {  
 $("#messageDiv").html(result);  
 },  
 error: function (xhr, status, error) {  
 $("#messageDiv").html(status);  
 }  
 });  
 }  
 return false;  
 });  
  
 function Validate() {  
 var errorCounter = 0;  
 $(".nameSpan").text("");  
 $(".ageSpan").text("");  
 $(".standardSpan").text("");  
 $(".percentSpan").text("");  
 $(".statusSpan").text("");  
  
 if ($("#name").val() == "") {  
 $(".nameSpan").text("Please enter name");  
 errorCounter++;  
 }  
  
 if ($("#age").val() == "") {  
 $(".ageSpan").text("Please enter age");  
 errorCounter++;  
 }  
 else if (!(isPositiveInteger($("#age").val()))) {  
 $(".ageSpan").text("Age should be in +ve integers");  
 errorCounter++;  
 }  
  
 if ($("#standard").val() == "") {  
 $(".standardSpan").text("Please enter standard");  
 errorCounter++;  
 }  
  
 if ($("#percent").val() == "") {  
 $(".percentSpan").text("Please enter percent");  
 errorCounter++;  
 }  
 else if (!(isPercent($("#percent").val()))) {  
 $(".percentSpan").text("Percent should be integers or decimal");  
 errorCounter++;  
 }  
  
 if ($("#status").val() == "Select") {  
 $(".statusSpan").text("Please select status");  
 errorCounter++;  
 }  
  
 if (errorCounter == 0) {  
 return true;  
 }  
 else {  
 return false;  
 }  
 }  
  
 function isPositiveInteger(n) {  
 return ($.isNumeric(n) && (n > 0));  
 }  
  
 function isPercent(n) {  
 return (n.match(/(^100(\.0{1,2})?$)|(^([1-9]([0-9])?|0)(\.[0-9]{1,2})?$)/));  
 }  
 });  
</script>

I passed “/index/insert” to the jQuery AJAX function to call the “insert” function of the “insert” controller. The C# function’s returned value is shown by the “success: function(){}” given inside the jQuery AJAX method.

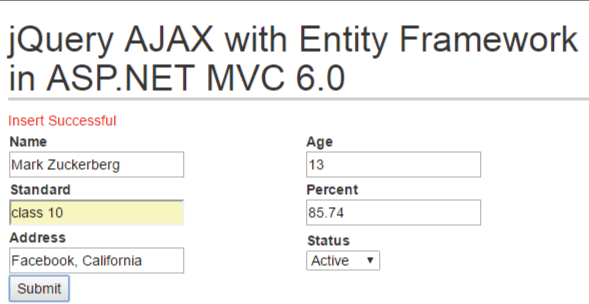
**STEP 8 — Configuring RouteConfig.cs**

To set “**Index**” view of the “Insert” controller as the starting page of our project, update the **RouteConfig.cs**file (inside the App\_Start folder) as shown below

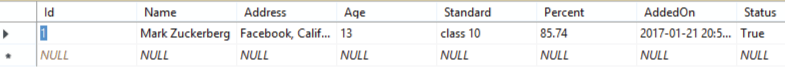
routes.IgnoreRoute("{resource}.axd/{\*pathInfo}");  
routes.MapRoute(  
 name: "Default",  
 url: "{controller}/{action}/{id}",  
 defaults: new { controller = "Insert", action = "Index", id = UrlParameter.Optional }  
);

**Testing**

To test it, let us fill and submit the form. You will see the **jQuery AJAX** will call the C# function, which in turn will insert the values and return a message. This return message will be shown in the div.



**FORM**



**Table**

**Summary**

This tutorial will help you to learn using **Entity Framework & jQuery AJAX** in ASP.NET MVC. Now you can go on and learn more about the Entity Framework.

**“I have also written a related tutorial which is about**[**Inserting Records in Entity Framework Core**](http://www.yogihosting.com/insert-records-entity-framework-core/)**, do check this too.”**

Please mark and vote for this tutorial and give me your suggestions.

Do follow me to learn more about ASP.NET web technology.